Fundamental Experiments with the Tucker Decomposition in the Matlab Tensor Toolbox

Sergio Garcia Tapia[1], Rebecca Hsu[2], Alyssa Hu[3], Darren Stevens II[4]

[1]Department of Mathematics, University at Buffalo, SUNY
[2]Department of Mathematics, University of Maryland, College Park
[3]Department of Mathematics and Department of Computer Science, University of Maryland, College Park
[4]Department of Computer Science and Electrical Engineering, UMBC

Advisor: Matthias K. Gobbert, gobbert@umbc.edu,
Department of Mathematics and Statistics, UMBC

Abstract

This report explores how data structures known as tensors can be used to perform multi-dimensional data analysis. If a matrix can be thought of as a two-dimensional array, then a tensor can be thought of as a multi-dimensional array (with more than two dimensions). Tensor decompositions are algorithms and tools that can allow the user to directly perform analysis on this type of data. After explaining the basics of tensors, we work with two different three-dimensional data sets and decompose the tensors in order to provide analysis and interpretations of various aspects of the data. We show in detail how to use commands from the Matlab Tensor Toolbox to set up the problems and compute the Tucker decomposition.

Key words. Tensors, Tucker tensors, Tucker decomposition, Matlab Tensor Toolbox, Principal component analysis.

1 Introduction

Many application problems in data analysis inherently contain multi-dimensional data. Potential examples can include behavioral studies across different situations, facial recognition algorithms, and signals processing and analysis. Matrices are oftentimes used to organize the variables reordered after having found an appropriate way to express the data in the two-dimensional array. On the other hand, one can also directly perform analysis on the data by using multi-dimensional arrays that preserve the inherent structure of such multi-dimensional data represented by tensors.

The basics of how to use and operate with tensors are similar to operations with regular two-dimensional matrices. With third-order (three-dimensional) tensors, it is clearer to visualize them as slices of matrices overlayed on top of each other. Multiplying tensors requires an extra step; one must unfold the tensor into a matrix before proceeding with traditional matrix multiplication and then refold the result back into a higher order tensor. For multi-dimensional data, one would normally perform dimension reduction techniques to organize the data into a matrix. Then, a method such as principal component analysis (PCA) would be applied on the matrix in order to provide analysis of the data. However, we can apply what is known as a tensor decomposition to all of the multi-dimensional data at once. One specific tensor decomposition for N-dimensional tensors is called the Tucker decomposition [5]; specifically for 3-dimensional data, Kiers et al. [4] also refer to it as 3-mode-PCA (3MPCA). This decomposition factors the original data into a core tensor that is typically smaller, as well as three factor matrices that measure the level of interaction

between the data [5]. Section 2 introduces some basic facts and properties of tensors and explains the Tucker decomposition more in-depth.

The Matlab Tensor Toolbox[1] has many functions available for creating and operating with tensors, some of which we will discuss in Section 3. The Tensor Toolbox can be used to perform basic operations on both dense and sparse tensors, such as set up and multiply tensors with other tensors and with other vectors or matrices. There are in fact functions for working with tensors that have specific properties, as well as relevant decompositions, such as Tucker or Kruskal tensors. Section 3 discusses the features of the Matlab Tensor Toolbox in more detail.

In order to provide an illustrative example, Kiers [4] provides a fictitious set of data as a behavioral study for six different individuals measuring five behavioral responses in four different scenarios. This data is naturally multi-dimensional and can be expressed as a $6 \times 5 \times 4$ tensor. Kiers applies the 3MPCA decomposition to this data to generate the component matrices and desired core tensor. Labels were given to each of the components after seeing the column groupings in order to provide a clear interpretation of the data. Kiers demonstrates mathematically how to use the results of the decomposition in order to retrieve the original data. This is shown in detail in Section 4.

Section 5 introduces data from a Dutch psychological experiment with a $326 \times 5 \times 2$ tensor. We apply the Tucker decomposition also to this data set and then use additional steps to analyze the result. Kiers used the component matrices' columns to group data and generated labels for those columns. For the Dutch data set, we take the dot product of the original data with the columns of one of the component matrices to better summarize the data. We note how we can input different core tensor sizes and if that affects resulting component matrices and interpretation.

Section 6 summarizes the conclusions from our calculations using the Matlab Tensor Toolbox and provides an outlook to larger problems.

2 Tensor Analysis

2.1 Tensor Basics

We now formally introduce tensors. Material presented here is based on the Kolda and Bader paper [5]. Recall that a matrix $U \in \mathbb{R}^{I \times J}$ can be thought of as a 2-dimensional array, where I, J are positive integers. A tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ is a generalization of a matrix; that is, it is an N-dimensional array. As such, many properties and operations of matrices are retained in higher dimensions with some concepts being more intricate. Tensors are denoted by Euler script characters as first done above in the usage of $\mathcal{X}$. An element in a tensor is denoted by $x_{i_1,\ldots,i_N}$, where $i_n = 1, \ldots, I_n$ (that is, the subscripts range from 1 to their corresponding uppercase equivalent). An N-dimensional tensor is also referred to sometimes as a tensor of order N, or a tensor with N modes. Addition is done element-wise as one would expect, and the Frobenius norm of a tensor is the same as that of a matrix, satisfying

$$||\mathcal{X}|| = \sqrt{\sum_{i_1=1}^{I_1} \cdots \sum_{i_N=1}^{I_N} x_{i_1,\ldots,i_N}^2} \quad \text{for } \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}.$$

[1] Version 2.6, http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.6.html

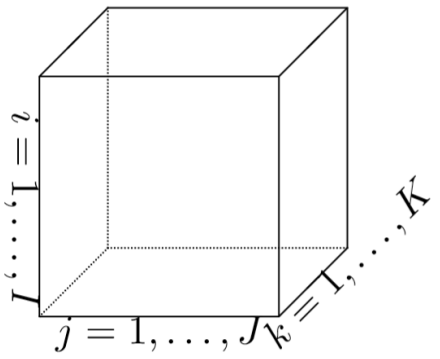


Figure 2.1: A tensor of order 3.

For N-way tensor $\mathcal{X}$, fibers and slices are defined by fixing $N-1$ and $N-2$ indices, respectively. The fibers then are 1-dimensional tensors or arrays (vectors), and the slices are 2-dimensional tensors or arrays (matrix) obtained from $\mathcal{X}$ with some orientation that depends on what indices have been fixed. For illustrative purposes, we limit our discussion to 3-way tensors and provide Figure 2.1 as a way to visualize them. For a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ of 3 dimensions, on the one hand, a *fiber* is obtained by fixing two of the indices. The possible choices then are vectors $\mathcal{X}(:, j, k)$, $\mathcal{X}(i, :, k)$, and $\mathcal{X}(i, j, :)$, for $i = 1, \ldots, I$, $j = 1, \ldots, J$, and $k = 1, \ldots, K$; these correspond to columns, rows, and tube fibers, respectively. In Figure 2.2, we see that fibers of a 3-way tensor are pillars with some orientation. On the other hand, a *slice* of a tensor $\mathcal{X}$ is obtained by fixing one of the indices and the choices would be matrices $\mathcal{X}(i, :, :)$, $\mathcal{X}(:, j, :)$, and $\mathcal{X}(:, :, k)$ that correspond to horizontal, lateral, and frontal slices, respectively. In Figure 2.3, we see that the the slices of a 3-way tensor are represented as planes with certain orientations.

Introducing fibers also allows us to introduce the idea of unfolding tensors. Visually, the mode-n *unfolding* or *matricization* of a 3-way tensor $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$, where $n = 1$, 2, or 3, means to take the mode-n fibers, to rotate them into column orientation, and then line them up sequentially. The result is that we express a tensor $\mathcal{G}$ as a matrix $G_{(n)}$ with dimensions that depend on $n = 1, 2, 3$, as well as P, Q, R. The benefit of this is that it allows us to consider mode-n matrix multiplication. The mode-n product between a tensor and matrix is defined if certain dimensions match as is the case for matrices. If $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is a tensor of 3 modes, and if $G_{(3)} \in \mathbb{R}^{R \times PQ}$ is its mode-3 unfolding, then the mode-3 product with a matrix $C \in \mathbb{R}^{K \times R}$ is expressed as

$$\mathcal{Y} = \mathcal{G} \times_3 C \iff Y_{(3)} = CG_{(3)}.$$

where $\mathcal{Y} \in \mathbb{R}^{P \times Q \times K}$, and $Y_{(3)} \in \mathbb{R}^{K \times PQ}$ is its mode-3 unfolding. An example of this is explored in detail in Section 4.

2.2 Principal Component Analysis

Many methods for dealing with multi-dimensional data by using matrices have been developed. Among them is principal component analysis (PCA), a technique that seeks to find an appropriate representation for data that has been collected and organized in a matrix. Generally, data is arranged in a matrix U such that each column constitutes a different type of measurement performed

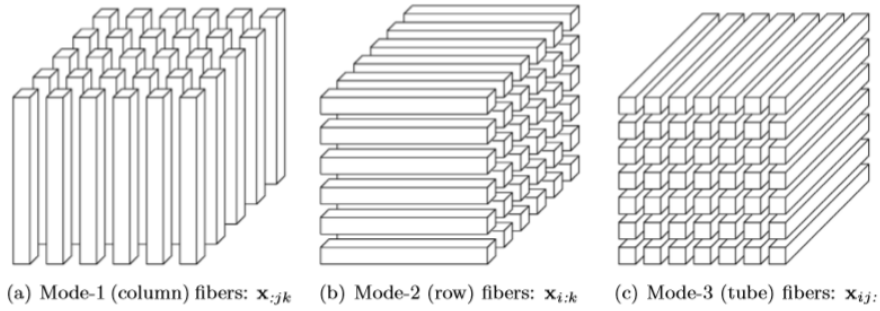


Figure 2.2: Fiber representations of a 3-way tensor.

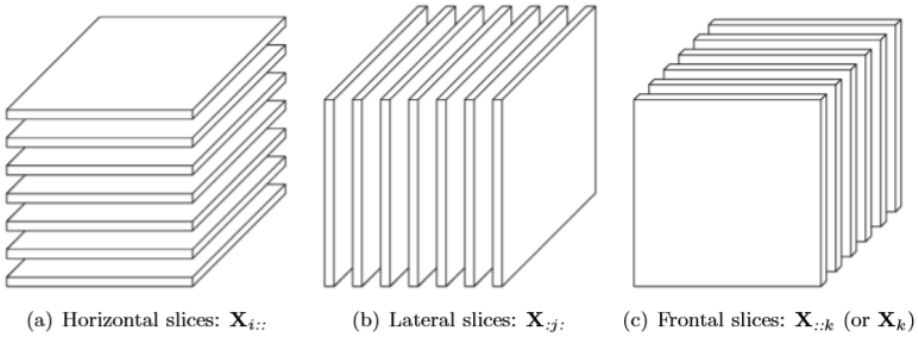


Figure 2.3: Slice representations of a 3-way tensor.

and each row a different trial. The goal is then to find a matrix P such that its vectors (known as the principal components of U) are a basis for the row space of U where the trial vectors are contained. Since each principal vector is associated with a corresponding variance between measurement types in A, the independent directions related to a large variance suggest underlying patterns in the data. A generalization of this technique to higher-order tensors has been developed and it is known as a Tucker decomposition [5] or 3MPCA [4]. The convenience is that we are then able to retain the multi-dimensional nature of data and perform analysis on it directly, instead of having to represent it as a matrix to use matrix analysis techniques.

2.3 Tucker Tensor Decomposition

We mentioned that tensors can be decomposed. Here, we concern ourselves with a method resembling PCA, known as the Tucker decomposition. Given a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, a Tucker decomposition attempts to express it as

$$\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C \tag{2.1}$$

for core tensor $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$, and with $A \in \mathbb{R}^{I \times P}$, $B \in \mathbb{R}^{J \times Q}$, and $C \in \mathbb{R}^{K \times R}$ being factor matrices that are orthogonal visually represented by Figure 2.4, for integers $1 \leq P \leq I$, $1 \leq Q \leq J$, and $1 \leq R \leq K$. The numbered subscripts indicate the mode of multiplication in which the product between the tensor $\mathcal{G}$ and the matrices are being multiplied that was discussed in Section 2.1. The entries themselves are given by

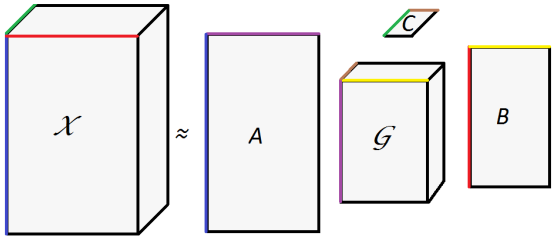


Figure 2.4: Visual representation of Tucker decomposition.

$$x_{ijk} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} a_{ip} b_{jq} c_{kr}.$$

By saying the decomposition resembles PCA, we mean that it is a form of higher-order principal component analysis where the component matrices are principal components [5], which is why Kiers [4] refers to it as 3MPCA. Therefore, we oftentimes use 3MPCA and Tucker decomposition interchangeably. The way in which to interpret the components of the decomposition is also sometimes dependent on the data set in question. Since the component matrices measure the level of interaction [5] between the information recorded, the technique can be used to group several variables that are most closely related to one another, as we demonstrate in Section 4. Alternatively, the decomposed result can be used jointly with the original data to provide insight into the recorded information, as shown in Section 5.

The Tucker decomposition for a given tensor is defined for different core tensor sizes, and therefore the decomposition is not unique. The decomposition can be computed through numerical methods such as Alternating Least Squares (ALS), which is an iterative method that depends on an initial guess, a stopping tolerance, and more, thus the result is not unique even for one choice of core tensor size.

3 Matlab Tensor Toolbox

The Tensor Toolbox has many functions available for creating and operating with tensors in Matlab. In particular, there are specific functions for working with tensors of specific structures, including regular tensors, sparse tensors, and Tucker-decomposed tensors.

3.1 Literature Review

The Tensor Toolbox for Matlab has been used and referenced in different research papers. Kolda and Bader's paper on Tensor Decompositions and Applications served as background reading for our research [5]. Other papers with interesting applications of the Tensor Toolbox will be briefly discussed. Fitzgerald, Coyle, and Cranitch investigate the separation of drum sounds from music with tensor factorization models [2]. Also, a lot of data can be represented with links and graphs, especially those involving relationships such as online connections in social networks. Thus, the Tensor Toolbox can be applied to predict future links, specifically by using a CP tensor decomposition (involving Alternating Least Squares) [1]. The Tensor Toolbox is compared to GigaTensor, an algorithm used to deal with large tensors and their decompositions, using Hadoop MapReduce [3].

Tensor Toolbox runs faster than GigaTensor when the size of the tensor (with same-size modes) is less than 10^7. However, any tensor of size greater than 10^7 and the Tensor Toolbox runs out of memory; thus, GigaTensor is helpful to use, as it appears to support tensors at least of size 10^9 [8]. Furthermore, Kolda and Sun previously commented on memory issues of applying Tucker on a large sparse tensor in a 2008 paper and propose a Memory Efficient Tucker method to avoid memory overflow [6].

3.2 Regular (Dense) Tensors

The `tensor` command is the most basic, which converts an existing array `A` into a tensor. Additionally, the user is allowed to pass an array argument specifying the desired size (e.g., [2 3 4] for a $2 \times 3 \times 4$ tensor). Alternatively, one can use the `tenrand` function to create a random tensor of a size determined by the passed array argument. Similar to functions `ones` and `zeroes` for matrices, the functions `tenones` and `tenzeros` create tensors of a desired size filled with ones and zeros, respectively. Having created a tensor, a user might be interested in information about it, such as its size or the information stored in it. Such inquiry can be done for a tensor `X` by `X.size` and `X.data` (or `double(X)`), with the former returning an array with the size and the latter a multi-dimensional array with the data that the tensor contains. In fact, using these two in conjunction is an easy way to replicate the tensor by passing the data array and size array. The extraction of elements is the same as for matrices, since one can just pass specific indices (or array of indices). Lastly, component-wise operations can be done with the dot operator, such as `X.*Y` for multiplication, or `X./Y` for division. Some functions, like the `sqrt` function for matrices, cannot be used; instead, the `tenfun(@foo, X)` is needed to perform the component-wise operations defined in some custom `@foo` function (note the character preceding the function name) to an existing tensor `X`. For instance, `tenfun(@sqrt,X)` takes the square root of every element in `X`.

3.3 Sparse Tensors

The syntax for the construction of sparse tensors is similar to that of regular tensors. The `sptensor` function is used, with the appropriate arguments being an array of subscripts (separated by a semicolon) for where to put non-zero entries, an array for the value desired at each subscript, and another to specify the size of the sparse tensor. If no size array argument is given, then the function will create a sparse tensor just big enough to hold the non-zero elements. Instead, one might be interested in using `sptenrand` to create a sparse tensor with random entries from a normal distribution on the interval (0,1), with some default mean and standard deviation. For this, one need only pass an array of the desired size, as well as an integer representing the number of non-zeros wanted on the sparse tensor. If, instead, such a number is not an integer and is instead a real number between the interval $(0, 1)$, then it is treated as a percentage of non-zeros in the sparse tensor (which, therefore, depends on size). Analogous to the case of normal tensors above, one can use `X.subs`, `X.vals`, and `X.size` to obtain the subscripts of the non-zero entries, the values of these entries, and the size of the sparse tensor, respectively. Another useful function is `find` (not limited to sparse tensors) which takes as input a tensor whose entries are subjected to a certain condition, such as `find(X>0.9)`, which considers the entries of tensor `X` that are greater than 0.9, and then extracts the corresponding subscripts. The `nnz` function also allows the user to obtain information on the tensor, in this case, the number of non-zeros. The `elemfun` function works for the non-zeros in sparse tensors the same way as the `tenfun` function works for regular tensors. For instance, to

obtain the square root for non-zero elements in a tensor `X`, one would do `elemfun(X, @sqrt)`.

3.4 Tucker Tensors

Another tensor construction that will be represented here is that of a Tucker tensor. Recall that a Tucker decomposition expresses a tensor of, say, order 3, as $\mathcal{X} \approx \mathcal{G} \times_1 A \times_2 B \times_3 C$ for core tensor $\mathcal{G}$ and matrices A, B, and C. One simply creates a tensor `G` with the tensor function, and a set of matrices as in `U = {A, B, C}` for matrices A, B, C. Then, passing these arguments in that order to the `ttensor` function creates the Tucker tensor, though only its components are output and not the full tensor. The components can be accessed by the dot operator, such as in `X.G` for the core tensor, and `U{i}` for the i-ith component matrix, $i = 1, 2, 3$. It is interesting to note that a Tucker tensor can be used as a core for another Tucker tensor.

Aside from being able to create a tensor whose Tucker decomposition components are pre-specified at the input phase, one can attempt to find the Tucker decomposition of a regular tensor by using the `tucker_als` function. The function takes a tensor and an array that specifies the best-rank approximation desired, so that it can output a Tucker-decomposed tensor by trying to fit an alternating least squares model. Other arguments the function can take (which have certain default values if not explicitly specified) are tolerance, the maximum number of iterations, and an initial guess. The idea is that if we know the rank R_n of the matrix obtained from each mode-n unfolding of the tensor and provide this as input, then the decomposition is more likely to be accurate.

3.5 Tensor Products and Miscellaneous

Tensor operations such as multiplication are handled with the `ttv`, `ttm`, and `ttt`. The first takes as arguments a tensor, a vector, and a number indicating the mode of multiplication (so that the product is defined). One can pass multiple vectors at once if enclosed in curly braces, such as `{A, B, C, D}`, followed by an array specifying the mode in which the vectors will be multiplied with the tensor. In the context above, it could be as `[1 3]`, which suggests the multiplications that will take place are the tensor with `A` in mode-1 and with `C` in mode-3, with the other multiplications not carried out since they are unspecified. The `ttm` multiplication works the same way. Lastly, the `ttt` function is used to multiply two tensors. One simply passes the two tensors as arguments and two arrays specifying which modes will be multiplied with each other (the order matters, for this might be the difference between the product being handled as an outer product or an inner product).

If at any point one wishes to unfold these tensors (that is, matricize them), one can do so with the `tenmat` function. Simply pass as arguments the tensor, `X`, and two arrays; one specifying the modes to be converted to rows, and the other specifying the modes to be converted to columns. Additionally, any type of tensor can be converted into a regular type of tensor via the `full` command, which takes as input any type of tensor and returns a dense (regular) tensor.

4 Illustrative Example of Tensor Decomposition

As discussed, the purpose of utilizing tensors and their decompositions is to make analyzing multi-dimensional data easier. Kiers et al. [4] provide several data sets that illustrate the Tucker decomposition and their respective interpretations, and they provide one particularly small illustrative

Table 4.1: Fictitious data set of scores of six individuals on five response variables for four situations.

Indi-	Doing an exam					Giving a speech					Family Picnic					Meeting a new date				
vidual	E	S	C	T	A	E	S	C	T	A	E	S	C	T	A	E	S	C	T	A
Anne	0.0	0.0	1.2	3.0	3.0	0.6	0.6	1.3	2.4	2.4	3.0	3.0	1.8	0.0	0.0	3.6	3.6	2.5	0.9	0.9
Bert	0.0	0.0	0.8	2.0	2.0	0.2	0.2	0.8	1.8	1.8	1.0	1.0	1.0	1.0	1.0	1.2	1.2	1.4	1.8	1.8
Claus	0.0	0.0	0.8	2.0	2.0	0.2	0.2	0.8	1.8	1.8	1.0	1.0	1.0	1.0	1.0	1.2	1.2	1.4	1.8	1.8
Dolly	0.0	0.0	1.2	3.0	3.0	0.6	0.6	1.3	2.4	2.4	3.0	3.0	1.8	0.0	0.0	3.6	3.6	2.5	0.9	0.9
Edna	0.0	0.0	1.0	2.5	2.5	0.4	0.4	1.1	2.1	2.1	2.0	2.0	1.4	0.5	0.5	2.4	2.4	2.0	1.3	1.3
Frances	0.0	0.0	1.2	3.0	3.0	0.6	0.6	1.3	2.4	2.4	3.0	3.0	1.8	0.0	0.0	3.6	3.6	2.5	0.9	0.9

example, for which the multiplication of the Tucker decomposition can be explicitly checked. Note that the article refers to this decomposition as 3MPCA and we will use this term while referring to tables from this article.

4.1 Original Input Data

Table 4.1 shows the original fictitious data provided in the Kiers article that will be used for the decomposition [4]. This table shows the scores of

- $I = 6$ individuals — Anne, Bert, Claus, Dolly, Edna, and Frances — displaying
- $J = 5$ behavior types — emotional (E), sensitive (S), caring (C), thorough (T), and accurate (A) — in
- $K = 4$ different situations — doing an exam, giving a speech, family picnic, and meeting a new date.

The data is presented in Table 4.1 in the form of the original (fictitious) psychological experiment. It gives naturally rise to a tensor of data $\mathcal{X} \in \mathbb{R}^{6\times5\times4}$, where the matrix of numbers in each subtable of the four situations is one slice $\mathcal{X}(:,:,k) \in \mathbb{R}^{6\times5}$ for each $k = 1,\ldots,4$, that is,

$$\mathcal{X}(:,:,1) = \begin{bmatrix} 0.0 & 0.0 & 1.2 & 3.0 & 3.0 \\ 0.0 & 0.0 & 0.8 & 2.0 & 2.0 \\ 0.0 & 0.0 & 0.8 & 2.0 & 2.0 \\ 0.0 & 0.0 & 1.2 & 3.0 & 3.0 \\ 0.0 & 0.0 & 1.0 & 2.5 & 2.5 \\ 0.0 & 0.0 & 1.2 & 3.0 & 3.0 \end{bmatrix}, \quad \mathcal{X}(:,:,2) = \begin{bmatrix} 0.6 & 0.6 & 1.3 & 2.4 & 2.4 \\ 0.2 & 0.2 & 0.8 & 1.8 & 1.8 \\ 0.2 & 0.2 & 0.8 & 1.8 & 1.8 \\ 0.6 & 0.6 & 1.3 & 2.4 & 2.4 \\ 0.4 & 0.4 & 1.1 & 2.1 & 2.1 \\ 0.6 & 0.6 & 1.3 & 2.4 & 2.4 \end{bmatrix},$$

$$\mathcal{X}(:,:,3) = \begin{bmatrix} 3.0 & 3.0 & 1.8 & 0.0 & 0.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 1.0 & 1.0 & 1.0 & 1.0 & 1.0 \\ 3.0 & 3.0 & 1.8 & 0.0 & 0.0 \\ 2.0 & 2.0 & 1.4 & 0.5 & 0.5 \\ 3.0 & 3.0 & 1.8 & 0.0 & 0.0 \end{bmatrix}, \quad \mathcal{X}(:,:,4) = \begin{bmatrix} 3.6 & 3.6 & 2.5 & 0.9 & 0.9 \\ 1.2 & 1.2 & 1.4 & 1.8 & 1.8 \\ 1.2 & 1.2 & 1.4 & 1.8 & 1.8 \\ 3.6 & 3.6 & 2.5 & 0.9 & 0.9 \\ 2.4 & 2.4 & 2.0 & 1.3 & 1.3 \\ 3.6 & 3.6 & 2.5 & 0.9 & 0.9 \end{bmatrix}.$$

To enter this data into Matlab, you first create a 3-D array `kiersX` and then use the function `tensor` from the Tensor Toolbox to build the tensor `tensorX`, as shown in the following code:

```
kiersX = zeros(6,5,4);
kiersX(:,:,1) = [0.0 0.0 1.2 3.0 3.0;
```

```
                0.0 0.0 0.8 2.0 2.0;
                0.0 0.0 0.8 2.0 2.0;
                0.0 0.0 1.2 3.0 3.0;
                0.0 0.0 1.0 2.5 2.5;
                0.0 0.0 1.2 3.0 3.0];
kiersX(:,:,2) = [0.6 0.6 1.3 2.4 2.4;
                0.2 0.2 0.8 1.8 1.8;
                0.2 0.2 0.8 1.8 1.8;
                0.6 0.6 1.3 2.4 2.4;
                0.4 0.4 1.1 2.1 2.1;
                0.6 0.6 1.3 2.4 2.4];
kiersX(:,:,3) = [3.0 3.0 1.8 0.0 0.0;
                1.0 1.0 1.0 1.0 1.0;
                1.0 1.0 1.0 1.0 1.0;
                3.0 3.0 1.8 0.0 0.0;
                2.0 2.0 1.4 0.5 0.5;
                3.0 3.0 1.8 0.0 0.0];
kiersX(:,:,4) = [3.6 3.6 2.5 0.9 0.9;
                1.2 1.2 1.4 1.8 1.8;
                1.2 1.2 1.4 1.8 1.8;
                3.6 3.6 2.5 0.9 0.9;
                2.4 2.4 2.0 1.3 1.3;
                3.6 3.6 2.5 0.9 0.9];
tensorX = tensor(kiersX)
```

Notice that the last line does not have a semi-colon, so Matlab will print out the tensor `tensorX`, which looks as follows and automatically includes its dimension:

```
tensorX is a tensor of size 6 x 5 x 4
      tensorX(:,:,1) =
               0         0    1.2000    3.0000    3.0000
               0         0    0.8000    2.0000    2.0000
               0         0    0.8000    2.0000    2.0000
               0         0    1.2000    3.0000    3.0000
               0         0    1.0000    2.5000    2.5000
               0         0    1.2000    3.0000    3.0000
      tensorX(:,:,2) =
          0.6000    0.6000    1.3000    2.4000    2.4000
          0.2000    0.2000    0.8000    1.8000    1.8000
          0.2000    0.2000    0.8000    1.8000    1.8000
          0.6000    0.6000    1.3000    2.4000    2.4000
          0.4000    0.4000    1.1000    2.1000    2.1000
          0.6000    0.6000    1.3000    2.4000    2.4000
      tensorX(:,:,3) =
          3.0000    3.0000    1.8000         0         0
          1.0000    1.0000    1.0000    1.0000    1.0000
          1.0000    1.0000    1.0000    1.0000    1.0000
```

```
    3.0000    3.0000    1.8000         0         0
    2.0000    2.0000    1.4000    0.5000    0.5000
    3.0000    3.0000    1.8000         0         0
tensorX(:,:,4) =
    3.6000    3.6000    2.5000    0.9000    0.9000
    1.2000    1.2000    1.4000    1.8000    1.8000
    1.2000    1.2000    1.4000    1.8000    1.8000
    3.6000    3.6000    2.5000    0.9000    0.9000
    2.4000    2.4000    2.0000    1.3000    1.3000
    3.6000    3.6000    2.5000    0.9000    0.9000
```

4.2 Illustrative 3MPCA Decomposition

Now, recall that a Tucker decomposition of a 3-way tensor $\mathcal{X}$ expresses the tensor as in (2.1). Since we specify in Matlab the size we want our core tensor to be, that will also determine the sizes of our component matrices. We specify the size based on how many classifications or groupings we want to use as our comparisons within each mode. For two groupings in each mode of $\mathcal{X} \in \mathbb{R}^{6\times5\times4}$, Kiers [4] chooses the core dimensions to be $2\times2\times2$ and then states one choice of possible component matrices

$$A = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \\ 0.5 & 0.5 \\ 1.0 & 0.0 \end{bmatrix} \in \mathbb{R}^{6\times2}, \quad B = \begin{bmatrix} 1.0 & 0.0 \\ 1.0 & 0.0 \\ 0.6 & 0.4 \\ 0.0 & 1.0 \\ 0.0 & 1.0 \end{bmatrix} \in \mathbb{R}^{5\times2}, \quad C = \begin{bmatrix} 1.0 & 0.0 \\ 0.8 & 0.2 \\ 0.0 & 1.0 \\ 0.3 & 1.2 \end{bmatrix} \in \mathbb{R}^{4\times2}, \tag{4.1}$$

and the faces of the core tensor $\mathcal{G} \in \mathbb{R}^{2\times2\times2}$

$$\mathcal{G}(:,:,1) = \begin{bmatrix} 0.0 & 3.0 \\ 0.0 & 2.0 \end{bmatrix}, \quad \mathcal{G}(:,:,2) = \begin{bmatrix} 3.0 & 0.0 \\ 1.0 & 1.0 \end{bmatrix}. \tag{4.2}$$

In A, rows 1, 4, and 6 are identical, and so are rows 2 and 3. In B, rows 1 and 2 are identical, as well as rows 4 and 5. In C, rows 1 and 2 are very similar, and so are rows 3 and 4. In terms of the application, the mathematical result of (4.1) is actually reported in [4] in the form of Table 4.2, where the classification into two groups in each component resulting from the core tensor of size $2 \times 2 \times 2$ are expressed in labels inspired by the application.

Table 4.2: Component values from 3MPCA.

Component matrix A			Component matrix B			Component matrix C		
Individual	Femininity	Masculinity	Response	Emotionality	Conscientiousness	Situation	Social	Performance
Anne	1.0	0.0	Emotional	1.0	0.0	Exam	0.0	1.0
Bert	0.0	1.0	Sensitive	1.0	0.0	Speech	0.2	0.8
Claus	0.0	1.0	Caring	0.6	0.4	Picnic	1.0	0.0
Dolly	1.0	0.0	Thorough	0.0	1.0	Date	1.2	0.3
Edna	0.5	0.5	Accurate	0.0	1.0			
Frances	1.0	0.0						

Table 4.3: Core $\mathcal{G} \in \mathbb{R}^{2\times2\times2}$ resulting from 3MPCA (Tucker decomposition).

	Performance Situations		Social Situations	
G: Core	Emotionality	Conscientiousness	Emotionality	Conscientiousness
Femininity	0.0	3.0	3.0	0.0
Masculinity	0.0	2.0	1.0	1.0

Each component shows the relationship between the elements within each mode and categorizes them based on the dimensions that we choose. For component A, we are simplifying our 6 people into 2 categories. Looking at the resulting matrix above, we can see that Anne, Dolly, and Frances have the same values, as well as Bert and Claus. From what Kiers tells us about these individuals [4], this split could be interpreted as grouping femininity against masculinity. Since Edna's original values lie between the masculine and feminine values, her component values are also half in each category and Kiers interprets this to mean that Edna is androgynous.

Component B shows the relationship between the qualities of behavior (Kiers refers to them as response variables). Notice how columns E and S of the original data match and columns T and A match. On the other hand, C is its own entity. Intuitively, we can say that emotional and sensitive behaviors could be grouped together and we could label that with Emotionality. We can also intuitively say that thorough and accurate behaviors could be grouped together, so we will label it as Conscientiousness. As shown in the Kiers paper, these two labels form the 2^{nd} mode. Caring does not neatly fit into either label, so as in the case with Edna, it exhibits a combination of the two labels.

The final component C shows the relationship between the different situations and groups them into two categories. Here, the data appears slightly harder to interpret since none of the entries match perfectly and cannot be grouped just based on that. Instead, we can interpret the entries here as part of each category with preferences towards one or another. From the labels Kiers used, the interpretation is that doing an exam is purely a performance situation, as opposed to a speech which is mostly performance but still has a social aspect. Looking at the original data, we can see that doing an exam and giving a speech have little or no scores in the E or S columns (social), with high scores in the T and A columns (performance). A picnic is an entirely social affair with almost no performance aspects, while a date is also a largely social affair that still has some performance aspects.

Table 4.3 again collects the faces of the core tensor from (4.2) in terms of the application. Based off our labels for the components, we use the same labels for the core. The core tensor gives a summary of all the interactions among the three sources of variation in the data. The front face $\mathcal{G}(:, :, 1)$ is related to performance situations, and the columns are labeled in [4] as conscientiousness and emotionality, respectively. The back face $\mathcal{G}(:, :, 2)$ on the other hand is related to social situations, and its columns are labeled as emotionality and conscientiousness, respectively. The first row of each face is related to femininity and the second row of each face is related to masculinity.

From the core tensor, we can summarize that females have very strong conscientiousness in performance situations and strong emotionality in social situations, but little emotionality in performance situations and low conscientiousness in social situations. This sounds intuitively correct. One might expect males to have similar conscientiousness in performance situations, but perhaps

less emotionality in social situations. We look back at the original data to see that females do have higher scores on the conscientiousness responses (T and A) in the performance situations (Exam, Speech) and higher scores on the emotionality responses (E and S) in the social situations (Picnic, Date). Also in the original data, we note that the two males have lower scores on the conscientiousness responses (T and A) in the performance situations (Exam, Speech) and lower scores for emotionality responses (E and S) for social situations (Picnic and Date).

4.3 Verifying and Understanding Results

For clarity, we explain how to carry out the computations necessary to obtain the tensor that corresponds to the Kiers table displayed earlier. First, the multiplication with the tensor that is taking place is dependent on the mode being considered. The idea is that a tensor has 3 dimensions, or modes, and subscripts indicate the mode of multiplication considered when taking the product with the factor matrices. For instance, mode-3 multiplication satisfies

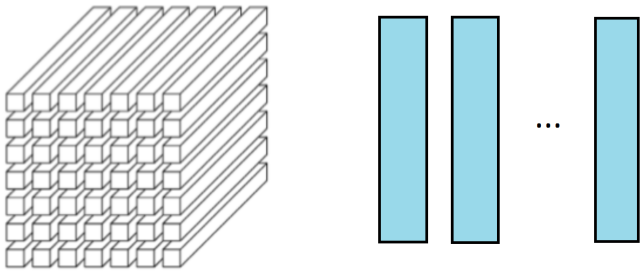
$$\mathcal{Y} = \mathcal{G} \times_3 C \iff Y_{(3)} = CG_{(3)}$$

where $C \in \mathbb{R}^{K\times R}$, the $G_{(3)} \in \mathbb{R}^{R\times PQ}$, and $Y_{(3)} \in \mathbb{R}^{K\times PQ}$ are mode-3 unfoldings of $\mathcal{G} \in \mathbb{R}^{P\times Q\times R}$ and $\mathcal{Y} \in \mathbb{R}^{P\times Q\times K}$. Hence, carrying out the mode-3 multiplication corresponds to unfolding or matricizing $\mathcal{G}$ and multiplying by C on the left (assuming the dimensions match for the product to be defined). Intuitively, unfolding a 3-way tensor $\mathcal{G}$ means to consider the fibers (that is, the columns, rows, or tubes) of a certain mode and making such fibers the columns of G_n (see Figure 2.2 for clarification of fibers). One can visualize this unfolding as taking the tube fibers, rotating them so as to make them columns, and then lining them up sequentially to obtain columns that form the tensor as a matrix; as presented in Figure 4.1. Thus, recall that the third component matrix in the decomposition mentioned in this section is

$$C = \begin{bmatrix} 1.0 & 0.0 \\ 0.8 & 0.2 \\ 0.0 & 1.0 \\ 0.3 & 1.2 \end{bmatrix}$$

and that the displayed frontal slices of core tensor $\mathcal{G} \in \mathbb{R}^{2\times 2\times 2}$ are

$$\mathcal{G}(:,:,1) = \begin{bmatrix} 0.0 & 3.0 \\ 0.0 & 2.0 \end{bmatrix}, \quad \mathcal{G}(:,:,2) = \begin{bmatrix} 3.0 & 0.0 \\ 1.0 & 1.0 \end{bmatrix}.$$



(c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Figure 4.1: Unfolding of mode-3 fibers.

Since the mode-3 unfolding means we make the tube fibers of $\mathcal{G}$ into columns of the unfolded matrix $G_{(3)}$, then tube fibers can be thought of as columns going into the page (towards the other slices), so that we get

$$G_{(3)} = \begin{bmatrix} 0.0 & 3.0 & 0.0 & 2.0 \\ 3.0 & 0.0 & 1.0 & 1.0 \end{bmatrix}$$

and the multiplication yields

$$Y_{(3)} = CG_{(3)} = \begin{bmatrix} 0.0 & 3.0 & 0.0 & 2.0 \\ 0.6 & 2.4 & 0.2 & 1.8 \\ 3.0 & 0.0 & 1.0 & 1.0 \\ 3.6 & 0.9 & 1.2 & 1.8 \end{bmatrix}.$$

We can then follow this procedure for the other modes by unfolding the tensor represented by these slices in the appropriate mode, so that one can carry out the multiplications in modes 1 and 2 with component matrices A and B, respectively.

The Matlab Tensor Toolbox can of course do these calculations for us, namely the function `ttm` implements the entire calculation $\mathcal{G} \times_1 A \times_2 B \times_3 C$ in (2.1). For the matrices A, B, C, and tensor $\mathcal{G}$ in (4.1)–(4.2), the Matlab code would be

```
A = [1.0 0.0;
     0.0 1.0;
     0.0 1.0;
     1.0 0.0;
     0.5 0.5;
     1.0 0.0];
B = [1.0 0.0;
     1.0 0.0;
     0.6 0.4;
     0.0 1.0;
     0.0 1.0];
C = [1.0 0.0;
     0.8 0.2;
     0.0 1.0;
     0.3 1.2];
G = zeros(2,2,2);
G(:,:,1) = [0.0 3.0;
            0.0 2.0];
G(:,:,2) = [3.0 0.0;
            1.0 1.0];
tensorG = tensor(G);
X = ttm(tensorG,{A,B,C})
```

which results in the output

```
X is a tensor of size 6 x 5 x 4
       X(:,:,1) =
                0         0    1.2000    3.0000    3.0000
```

```
         0         0    0.8000    2.0000    2.0000
         0         0    0.8000    2.0000    2.0000
         0         0    1.2000    3.0000    3.0000
         0         0    1.0000    2.5000    2.5000
         0         0    1.2000    3.0000    3.0000
X(:,:,2) =
    0.6000    0.6000    1.3200    2.4000    2.4000
    0.2000    0.2000    0.8400    1.8000    1.8000
    0.2000    0.2000    0.8400    1.8000    1.8000
    0.6000    0.6000    1.3200    2.4000    2.4000
    0.4000    0.4000    1.0800    2.1000    2.1000
    0.6000    0.6000    1.3200    2.4000    2.4000
X(:,:,3) =
    3.0000    3.0000    1.8000         0         0
    1.0000    1.0000    1.0000    1.0000    1.0000
    1.0000    1.0000    1.0000    1.0000    1.0000
    3.0000    3.0000    1.8000         0         0
    2.0000    2.0000    1.4000    0.5000    0.5000
    3.0000    3.0000    1.8000         0         0
X(:,:,4) =
    3.6000    3.6000    2.5200    0.9000    0.9000
    1.2000    1.2000    1.4400    1.8000    1.8000
    1.2000    1.2000    1.4400    1.8000    1.8000
    3.6000    3.6000    2.5200    0.9000    0.9000
    2.4000    2.4000    1.9800    1.3500    1.3500
    3.6000    3.6000    2.5200    0.9000    0.9000
```

This result confirms indeed that for this small case with explicitly chosen numbers, the product $\mathcal{G} \times_1 A \times_2 B \times_3 C$ is nearly equal to the input tensor. That is, the tensor `X` calculated by `ttm` above is different from the original tensor `tensorX` in Section 4.1 in only a few components. Recall that these tensors have dimension $6 \times 5 \times 4$, but `X` is here calculated from a core tensor with lower dimensions, so it is to be expected that the agreement is not perfect even for such a small data set.

The matrices in this section were not calculated, but chosen by hand in Kiers [4]. Their product is nearly equal to the original tensor, which is not common, in particular for a core tensor with smaller dimensions than the original data tensor. So, the data and decomposition in this section provides an example of (2.1) with nearly equality instead of approximation.

4.4 Computed Tucker Decomposition

Ordinarily, it is not possible to guess the values of A, B, C, and $\mathcal{G}$ in the Tucker decomposition, of course. We demonstrate here how to use the Matlab Tensor Toolbox function `tucker_als` to compute one decomposition. Namely, for the tensor $\mathcal{X} \in \mathbb{R}^{6\times5\times4}$ and requesting a core tensor with dimensions $2 \times 2 \times 2$, the Matlab code after setting up the tensor `tensorX` as in Section 4.1 would read

```
tuckerX = tucker_als(tensorX, [2 2 2])
```

The output variable `tuckerX` is a struct that holds A, B, C, and $\mathcal{G}$ as fields `U{1}`, `U{2}`, `U{3}`, and `core`, respectively, thus the following Matlab code can be used to extract them:

```
A = tuckerX.U{1}
B = tuckerX.U{2}
C = tuckerX.U{3}
tensorG = tuckerX.core
```

Back in mathematical notation, the obtained matrices $A \in \mathbb{R}^{6\times 2}$, $B \in \mathbb{R}^{5\times 2}$, $C \in \mathbb{R}^{4\times 2}$ are

$$A = \begin{bmatrix} 0.4869 & -0.2690 \\ 0.2697 & 0.6151 \\ 0.2697 & 0.6151 \\ 0.4869 & -0.2690 \\ 0.3786 & 0.1617 \\ 0.4869 & -0.2690 \end{bmatrix}, \quad B = \begin{bmatrix} 0.4615 & -0.4646 \\ 0.4615 & -0.4646 \\ 0.4472 & -0.0678 \\ 0.4324 & 0.5309 \\ 0.4324 & 0.5309 \end{bmatrix}, \quad C = \begin{bmatrix} 0.3844 & 0.6853 \\ 0.4013 & 0.4577 \\ 0.4746 & -0.4531 \\ 0.6826 & -0.3400 \end{bmatrix},$$

and faces of the core tensor $\mathcal{G} \in \mathbb{R}^{2\times 2\times 2}$ are

$$\mathcal{G}(:,:,1) = \begin{bmatrix} 16.3956 & -0.5072 \\ 0.8613 & 3.4340 \end{bmatrix} \quad \mathcal{G}(:,:,2) = \begin{bmatrix} 0.5610 & 9.9596 \\ 0.1826 & -1.255 \end{bmatrix}.$$

As discussed at the end of Section 2.1 already, a Tucker decomposition is not unique, and moreover, `tucker_als` is an iterative method, so different runs can give different results.

We will assume that the labels for the columns in the component matrices are the same as in the Kiers article [4]. We can see that the females have the same values and the males have the same values, with only Edna being different. It also makes sense that females have larger values of femininity as opposed to masculinity (note the negative signs for masculinity), though it is surprising that the males have much higher values of masculinity, but still some femininity (note the positive signs). We can see something similar with the grouping of the response variables. From this, we might also interpret that an exam is mostly a performance situation, with still some social aspects which sounds unintuitive.

Looking at the core tensor, we decided to reverse the labeling for the columns in order to preserve the interpretation under the assumption that the high value of 16.3956 corresponds to what should be the high value for female conscientiousness in performance situations. Just as in the example from Kiers, we can summarize that females have very strong conscientiousness in performance situations and strong emotionality in social situations, but little emotionality in performance situations and low conscientiousness in social situations, which is similar to results in the Kiers article. One might also expect males to have similar conscientiousness in performance situations, but perhaps less emotionality in social situations. Surprisingly, this decomposition tells us that males have a fairly strong emotionality towards performance situations, with little response at all towards social scenarios. Looking back at the data, we can see that females do indeed have higher scores on the conscientiousness responses (T and A) and higher scores on the emotionality responses (E and S) in the performance situations (Exam, Speech) and social situations (Picnic, Date) respectively. While the two males do show low scores on everything in the social situations, they don't show comparatively higher emotionality on the performance situations. This iterative process leads to a different interpretation than the Kiers article and is a potential nuance for further study.

Table 4.4: Component matrix A for the Tucker decomposition with 3 categories.

Component matrix A			
Individual	Femininity	Masculinity	Other
Anne	0.4869	−0.2690	−0.1546
Bert	0.2697	0.6151	−0.2211
Claus	0.2697	0.6151	−0.2211
Dolly	0.4869	−0.2690	−0.1546
Edna	0.3786	0.1615	0.9114
Frances	0.4869	−0.2690	−0.1546

4.5 Other Core Tensor Dimensions

Since the Tucker decomposition allows us to specify the dimensions we want to use, we are not limited to cube cores (such as $2 \times 2 \times 2$). Say, for instance, we wanted to see if using 3 classifications for people beyond male and female would be a better fit to represent Edna as opposed to being a composite between male and female. We would specify the core tensor to be $3 \times 2 \times 2$, and the first component should have 3 columns. The individuals who are purely one or the other should have the same if not very similar values in their original columns, while Edna should undergo a drastic change in both of the 2 original columns now that Edna is freed from the constraint of 2 classifications.

Comparing the A matrix from Section 4.4 to Table 4.4, we see that our original columns for everyone except Edna are the same, and everyone has negative values in the third column, except for Edna in this instance. If the Kiers article did something similar, we could expect the third column to have a 1.0 for Edna with 0.0 in masculinity and femininity. A similar component matrix could be generated for the response variables. Since Caring has a different value from all other response variables, it may have a strong correlation to the third category if we specified our core to have dimensions of $2 \times 3 \times 2$. It should be noted that having core dimensions very close to the original data's dimensions serve little purpose since the goal is to shrink the original data to a size that can be summarized. Having any of the core dimensions as 1 would also be unproductive, since we would not be categorizing the modes or showing any useful numbers for comparison.

5 Practical Application of Tensor Decomposition

We now look at a larger data set that still has the same structure as the illustrative example in Section 4. This data set is one of many available through the Three-Mode Company [7], a website devoted to providing three-mode data sets to motivate the use of three-mode data analysis. We discuss a couple ways in which one can make use of the results of the decomposition to obtain meaningful information.

5.1 Dutch Children Psychological Experiment

The data set consists of results from a Dutch psychological experiment involving:

- $I = 326$ children who exhibited

Table 5.1: First five rows of Dutch children data.

Individual	PS	CM	R	AV	DI
1	3	2	1	2	7
2	6	7	1	1	1
3	1	2	1	2	7
4	7	7	7	1	1
5	6	4	4	1	1

- $J = 5$ behaviors — Proximity Seeking (PS), Contact Maintaining (CM), Resistance (R), Avoidance (AV), Distance Interaction (DI) — in
- $K = 2$ stressful situations where the children were put in a room with a stranger and the mother was later brought back.

Hence, the data can be represented in the form of a 3-way tensor $\mathcal{X} \in \mathbb{R}^{326\times5\times2}$. A score between 1 and 7 is given to each child to rate their behavior in each of the 5 categorical scales. Data collected for the first 5 children in situation 1 is displayed in Table 5.1 for example.

5.2 Applying Tucker Decompositions

We compute a Tucker decomposition using `tucker_als` with a requested core tensor size of $2\times2\times2$. The 2 for the first mode attempts to split the children in 2 groups. For the second mode, the 2 attempts to place the more closely related behaviors together. For the third mode, we pick it as a default since there are 2 situations. Thus, we obtain core $\mathcal{G} \in \mathbb{R}^{2\times2\times2}$, and component matrices $A \in \mathbb{R}^{326\times2}$, $B \in \mathbb{R}^{5\times2}$, and $C \in \mathbb{R}^{2\times2}$ as

$$A = \begin{bmatrix} 0.0476 & -0.0663 \\ 0.0571 & 0.0811 \\ \vdots & \vdots \\ 0.0556 & -0.0236 \\ 0.0547 & -0.0455 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5444 & -0.3705 \\ 0.4363 & -0.5090 \\ 0.3391 & -0.1313 \\ 0.3919 & 0.3124 \\ 0.4947 & 0.6992 \end{bmatrix}, \quad C = \begin{bmatrix} 0.7342 & 0.6789 \\ -0.6778 & 0.7342 \end{bmatrix},$$

$$\mathcal{G}(:,:,1) = \begin{bmatrix} 0.3376 & 16.3568 \\ -1.7602 & 0.7655 \end{bmatrix} \quad \mathcal{G}(:,:,2) = \begin{bmatrix} 178.6889 & -0.6780 \\ -0.0978 & -80.4707 \end{bmatrix}.$$

Oftentimes, the interest in psychological experiments is to understand behavioral patterns. With this motivation, we restrict our attention to the B matrix because it corresponds to the 5 response variables:

$$B = \begin{bmatrix} 0.5444 & -0.3705 \\ 0.4363 & -0.5090 \\ 0.3391 & -0.1313 \\ 0.3919 & 0.3124 \\ 0.4947 & 0.6992 \end{bmatrix} \begin{matrix} \text{Proximity Seeking} \\ \text{Contact Maintaining} \\ \text{Resistance} \\ \text{Avoidance} \\ \text{Distance Interaction} \end{matrix}$$

By sign and magnitude, the second column of B noticeably groups the first three behaviors of Proximity Seeking, Contact Maintaining, and Resistance and the last two behaviors of Avoidance and Distance Interaction.

We project the first 5 children's data onto the second column of B. In other words, we take the dot product of each row of the children's data with the second column of B and get

$$\mathcal{X}(1{:}5,:,1)\,B(:,2) = \begin{bmatrix} 3 & 2 & 1 & 2 & 7 \\ 6 & 7 & 1 & 1 & 1 \\ 1 & 2 & 1 & 2 & 7 \\ 7 & 7 & 7 & 1 & 1 \\ 6 & 4 & 4 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.3705 \\ -0.5090 \\ -0.1313 \\ 0.3124 \\ 0.6992 \end{bmatrix} = \begin{bmatrix} 3.2583 \\ -4.0955 \\ 3.9993 \\ -6.0638 \\ -3.7725 \end{bmatrix}.$$

Negative values correspond to the extent to which behaviors of Proximity Seeking and/or Contact Maintaining are present. Positive values correspond to the extent to which behaviors of Avoidance and/or Distance Interaction are present. This projection helps summarize information about the behavior of each child in situation 1.

The Tensor Toolbox allows us to request different tensor core sizes. For further exploration, we compute a Tucker decomposition by requesting a core tensor of size $2 \times 3 \times 2$, which results in component matrices $A \in \mathbb{R}^{326\times 2}$, $B \in \mathbb{R}^{5\times 3}$, $C \in \mathbb{R}^{2\times 2}$, and core tensor $\mathcal{G} \in \mathbb{R}^{2\times 3\times 2}$

$$A = \begin{bmatrix} 0.0476 & -0.0667 \\ 0.0570 & 0.0807 \\ \vdots & \vdots \\ 0.0552 & -0.0240 \\ 0.0546 & -0.0458 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5444 & -0.3706 & 0.6470 \\ 0.4363 & -0.5084 & -0.5573 \\ 0.3391 & -0.1316 & -0.3101 \\ 0.3919 & 0.3111 & 0.3238 \\ 0.4947 & 0.7001 & -0.2643 \end{bmatrix}, \quad C = \begin{bmatrix} 0.7342 & 0.6790 \\ -0.6790 & 0.7342 \end{bmatrix},$$

$$\mathcal{G}(:,:,1) = \begin{bmatrix} 0.3369 & 16.3405 & 4.6376 \\ -1.7663 & 0.7502 & 1.3683 \end{bmatrix}, \quad \mathcal{G}(:,:,2) = \begin{bmatrix} 178.6895 & -0.0863 & 0.0055 \\ -0.1397 & -80.4702 & 0.9545 \end{bmatrix}.$$

We also compute a Tucker decomposition for a core tensor of size $2\times 4\times 2$, which gives component matrices $A \in \mathbb{R}^{326\times 2}$, $B \in \mathbb{R}^{5\times 4}$, $C \in \mathbb{R}^{2\times 2}$, and core tensor $\mathcal{G} \in \mathbb{R}^{2\times 4\times 2}$

$$A = \begin{bmatrix} 0.0476 & -0.0663 \\ 0.0570 & 0.0809 \\ \vdots & \vdots \\ 0.0552 & -0.0242 \\ 0.0546 & -0.0456 \end{bmatrix}, \quad B = \begin{bmatrix} 0.5444 & -0.3713 & 0.6499 & -0.0087 \\ 0.4363 & -0.5086 & -0.5581 & -0.4555 \\ 0.3391 & -0.1305 & -0.3113 & 0.8760 \\ 0.3919 & 0.3128 & 0.3178 & -0.0483 \\ 0.4947 & 0.6990 & -0.2612 & -0.1509 \end{bmatrix}, \quad C = \begin{bmatrix} 0.7342 & 0.6789 \\ -0.6789 & 0.7342 \end{bmatrix},$$

$$\mathcal{G}(:,:,1) = \begin{bmatrix} 0.3446 & 16.3399 & 4.6349 & -0.4621 \\ -1.7529 & 0.7657 & 1.3615 & 1.6279 \end{bmatrix}, \quad \mathcal{G}(:,:,2) = \begin{bmatrix} 178.6896 & -0.0853 & 0.0051 & 0.0168 \\ -0.1362 & -80.4694 & 0.9541 & -0.0784 \end{bmatrix}.$$

In this instance, the first and second columns of B do not drastically change across the different sizes of the core tensor.

It is important to note at this point that we have performed a similar decomposition on two different sets of data, but used them for different methods of interpretation. With the Kiers article, the decomposition was used to show how all of the components related to each other and is an illustrative example on how to label the columns given that the properties of the individuals are known. For the Dutch experiment, we chose to only focus on the B matrix that corresponded to the response variables. Due to the similarity of sign and magnitude of the first column in B, we chose to focus on the second column. We have extended the analysis done with Tucker by projecting the rows of the original data onto the second column of the second component matrix B. Doing this allowed us to provide a summary of the children's behavioral tendencies. How one chooses to analyze and interpret the results depends on the data that is being used as well as what the user wants to look for.

6 Conclusions

Data can exist in more than two dimensions and tensor decompositions allow us to directly perform analysis on this multi-dimensional data, as opposed to performing multiple analyses on two-dimensional data through techniques such as principal component analysis (PCA). We considered two examples from psychological studies due to their intuitive nature, one small and one modest sized one. We showed the details of commands from the Matlab Tensor Toolbox needed to set up and analyze them.

The Matlab Tensor Toolbox (version 2.6) was able to do the calculations for the small amount of data in the examples of Sections 4 and 5 in seconds. This suggests it might do well with larger data sets, and highlights that other programming languages like C might be able to handle even larger computations. Using these tools, there is a wide range of potential applications for multi-dimensional data such as signals analytics, facial recognition, big-data analysis, etc. with much larger datasets.

Acknowledgments

These results were obtained as part of the REU Site: Interdisciplinary Program in High Performance Computing (hpcreu.umbc.edu) in the Department of Mathematics and Statistics at the University of Maryland, Baltimore County (UMBC) in Summer 2016. This program is funded by the National Science Foundation (NSF), the National Security Agency (NSA), and the Department of Defense (DOD), with additional support from UMBC, the Department of Mathematics and Statistics, the Center for Interdisciplinary Research and Consulting (CIRC), and the UMBC High Performance Computing Facility (HPCF). HPCF is supported by the U.S. National Science Foundation through the MRI program (grant nos. CNS–0821258 and CNS–1228778) and the SCREMS program (grant no. DMS–0821311), with additional substantial support from UMBC. Co-author Darren Stevens II was supported, in part, by the UMBC National Security Agency (NSA) Scholars Program through a contract with the NSA. The authors thank both our team's graduate assistant Jonathan Graf and faculty mentor Dr. Matthias K. Gobbert for their support throughout the program and beyond, and we are grateful to the project client Dr. Tyler Simon from the Laboratory for Physical Sciences for pointing us to the interesting issue of tensor analysis and the Matlab Tensor Toolbox.

References

[1] E. Acar, D. M. Dunlavy, and T. G. Kolda, *Link prediction on evolving data using matrix and tensor factorizations*, in 2009 IEEE International Conference on Data Mining Workshops, Dec. 2009, pp. 262–269.

[2] D. Fitzgerald, E. Coyle, and M. Cranitch, *Using tensor factorisation models to separate drums from polyphonic music*, in Proceedings of the International Conference on Digital Audio Effects (DAFX09), 2009.

[3] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, *Gigatensor: Scaling tensor analysis up by 100 times — algorithms and discoveries*, in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, ACM, 2012, pp. 316–324.

[4] H. A. L. Kiers and I. V. Mechelen, *Three-way component analysis: Principles and illustrative application*, Psychological Methods, 6 (2001), pp. 84–110.

[5] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM Review, 51 (2009), pp. 455–500.

[6] T. G. Kolda and J. Sun, *Scalable tensor decompositions for multi-aspect data mining*, in 2008 Eighth IEEE International Conference on Data Mining, Dec. 2008, pp. 363–372.

[7] P. M. Kroonenberg, *The Three-Mode Company*. http://three-mode.leidenuniv.nl. Accessed September 12, 2016.

[8] E. E. Papalexakis, U. Kang, C. Faloutsos, N. D. Sidiropoulos, and A. Harpale, *Large scale tensor decompositions: Algorithmic developments and applications*, IEEE Data Engineering Bulletin, 36 (2013), pp. 59–66.